CSC3210 Assignment-3

1. Using a table similar to that shown in *Figure 3.6*, calculate the product of the octal unsigned 6-bit integers 62 and 12 using the hardware described in *Figure 3.3*. You should show the contents of each register on each step. [1*30 = 30 POINTS]

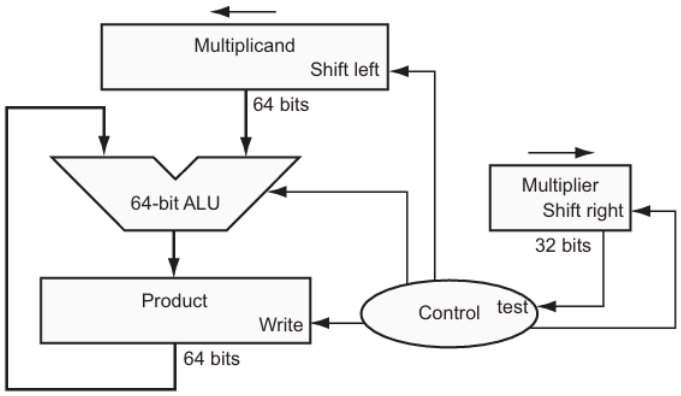

FIGURE 3.3 First version of the multiplication hardware. The Multiplicand register, ALU, and Product register are all 64 bits wide, with only the Multiplier register containing 32 bits. (Appendix A describes ALUs.) The 32-bit multiplicand starts in the right half of the Multiplicand register and is shifted left 1 bit on each step. The multiplier is shifted in the opposite direction at each step. The algorithm starts with the product initialized to 0. Control decides when to shift the Multiplicand and Multiplier registers and when to write new values into the Product register.

Iteration	Step	Multiplier	Multiplicand	Product
0	Initial values	001(1)	0000 0010	0000 0000
1	1a: 1 ⇒ Prod = Prod + Mcand	0011	0000 0010	0000 0010
	2: Shift left Multiplicand	0011	0000 0100	0000 0010
	3: Shift right Multiplier	000(1)	0000 0100	0000 0010
2	1a: 1 ⇒ Prod = Prod + Mcand	0001	0000 0100	0000 0110
	2: Shift left Multiplicand	0001	0000 1000	0000 0110
	3: Shift right Multiplier	000(0)	0000 1000	0000 0110
3	1: 0 ⇒ No operation	0000	0000 1000	0000 0110
	2: Shift left Multiplicand	0000	0001 0000	0000 0110
	3: Shift right Multiplier	000(0)	0001 0000	0000 0110
4	1: 0 ⇒ No operation	0000	0001 0000	0000 0110
	2: Shift left Multiplicand	0000	0010 0000	0000 0110
	3: Shift right Multiplier	0000	0010 0000	0000 0110

FIGURE 3.6 Multiply example using algorithm in Figure 3.4. The bit examined to determine the next step is circled in color.

2. As discussed in the text, one possible performance enhancement is to do a shift and add instead of an actual multiplication. Since 9 × 6, for example, can be written (2 × 2 × 2 + 1) × 6, we can calculate 9 × 6 by shifting 6 to the left three times and then adding 6 to that result.

Show the best way to calculate 0x33 × 0x55 using shifts and adds/subtracts. Assume both inputs are 8-bit unsigned integers. [1*10 = 10 POINTS]

NOTE: "0x" specifies a hexadecimal value, thus asking you to multiply 33 (hexadecimal) by 55 (hexadecimal).

3. Using a table similar to that shown in *Figure 3.10*, calculate 74 divided by 21 using the hardware described in *Figure 3.8*. You should show the contents of each register on each step. Assume both inputs are unsigned 6-bit integers. [1*30 = 30 POINTS]

   NOTE: Because 74 is a 7-bit value, you should assume that it should be 74(octal) and 21(octal).

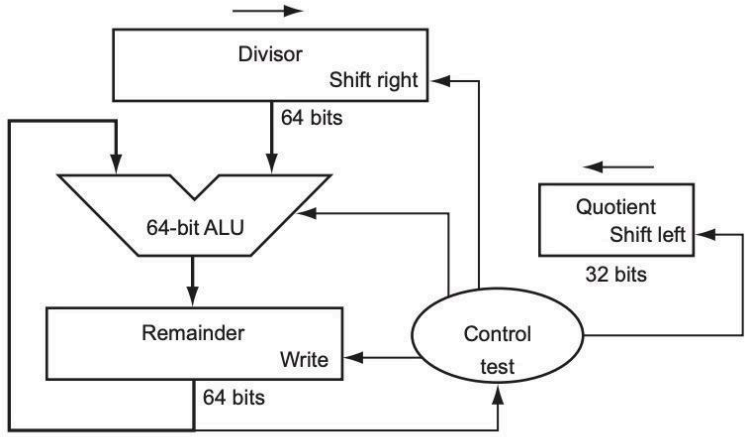

FIGURE 3.8 First version of the division hardware. The Divisor register, ALU, and Remainder register are all 64 bits wide, with only the Quotient register being 32 bits. The 32-bit divisor starts in the left half of the Divisor register and is shifted right 1 bit each iteration. The remainder is initialized with the dividend. Control decides when to shift the Divisor and Quotient registers and when to write the new value into the Remainder register.

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem – Div	0000	0010 0000	(1)110 0111
	2b: Rem < 0 ⇒ +Div, SLL Q, Q0 = 0	0000	0010 0000	0000 0111
	3: Shift Div right	0000	0001 0000	0000 0111
2	1: Rem = Rem – Div	0000	0001 0000	(1)111 0111
	2b: Rem < 0 ⇒ +Div, SLL Q, Q0 = 0	0000	0001 0000	0000 0111
	3: Shift Div right	0000	0000 1000	0000 0111
3	1: Rem = Rem – Div	0000	0000 1000	(1)111 1111
	2b: Rem < 0 ⇒ +Div, SLL Q, Q0 = 0	0000	0000 1000	0000 0111
	3: Shift Div right	0000	0000 0100	0000 0111
4	1: Rem = Rem – Div	0000	0000 0100	(0)000 0011
	2a: Rem ≥ 0 ⇒ SLL Q, Q0 = 1	0001	0000 0100	0000 0011
	3: Shift Div right	0001	0000 0010	0000 0011
5	1: Rem = Rem – Div	0001	0000 0010	(0)000 0001
	2a: Rem ≥ 0 ⇒ SLL Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

FIGURE 3.10 Division example using the algorithm in Figure 3.9. The bit examined to determine the next step is circled in color.

4. What decimal number does the bit pattern 0x0C000000 represent if it is a floating point number? Use the IEEE 754 standard. [1*15 = 15 POINTS]

5. Write down the binary representation of the decimal number 63.25 assuming the IEEE 754 double precision format. [1*15 = 15 POINTS]

NOTE:

- An explanation is expected with every answer and may count for half of the points or more. If your answer does not have an explanation, or that explanation is incomplete, expect to lose points. This is because we want to see how you arrive at the answer.
- For this homework, answer the following questions from the book(please see the .pdf file):
- These problems can be found in section 3.13 of your book (starting on page 246 of the physical copy: Chapter 3, exercises 3.12, 3.17, 3.18, 3.22, 3.24).
- Since you are answering problems from the book, make a .pdf file like you do for the lab reports. It should have its own cover page. However, you do not need the different sections (e.g. introduction) that the lab reports have.

- As you are aware, certain types of assistance are inappropriate in this class. Read the collaboration policy given to you at the beginning of the semester, if you have any questions.